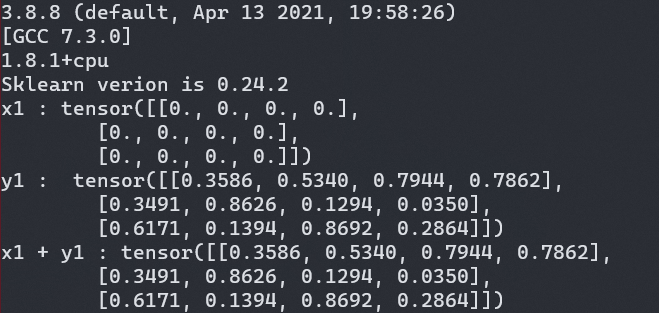
**2021年安徽省大数据与人工智能应用竞赛**

**人工智能赛道-网络赛答题卡（本科组）**

**学校名称： 合肥工业大学**  **队伍名称： 物联网小分队 登录账号：ahaib066**

**第一部分：人工智能基础环境搭建部署**（15分）



**第二部分：样本数据预处理**（30分）

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler,StandardScaler

df=pd.read\_csv('task2.csv')

data=np.array(df)

max\_Temp=data[:,1]

min\_Temp=data[:,3]

Mean\_Temp=data[:,2]

max\_Temp\_mean=np.mean(max\_Temp)

max\_Temp\_var=np.var(max\_Temp)

min\_Temp\_mean=np.mean(min\_Temp)

min\_Temp\_var=np.var(min\_Temp)

Mean\_Temp\_Max=np.max(Mean\_Temp)

Mean\_Temp\_Min=np.min(Mean\_Temp)

print("max\_Temp均值:",max\_Temp\_mean)

print("max\_Temp差:",max\_Temp\_var)

print("min\_Temp均值:",min\_Temp\_mean)

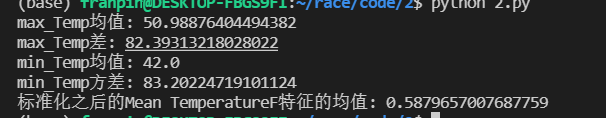
print("min\_Temp方差:",min\_Temp\_var)

Mean\_Temp\_Std=[(x-Mean\_Temp\_Min)/(Mean\_Temp\_Max-Mean\_Temp\_Min) for x in Mean\_Temp]

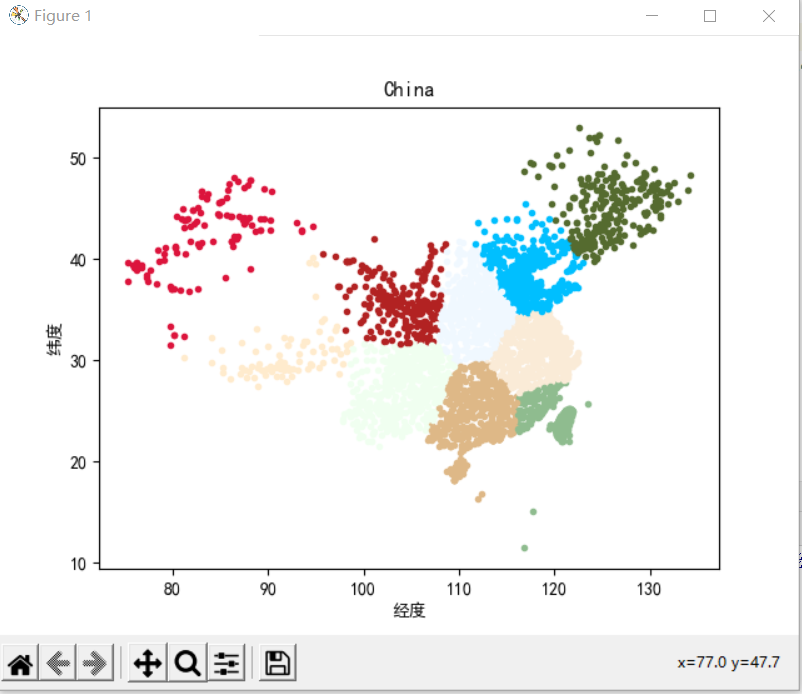
Mean\_Temp\_Std=np.array(Mean\_Temp\_Std)

Mean\_Temp\_Std\_mean=np.mean(Mean\_Temp\_Std)

print("标准化之后的Mean TemperatureF特征的均值:",Mean\_Temp\_Std\_mean)



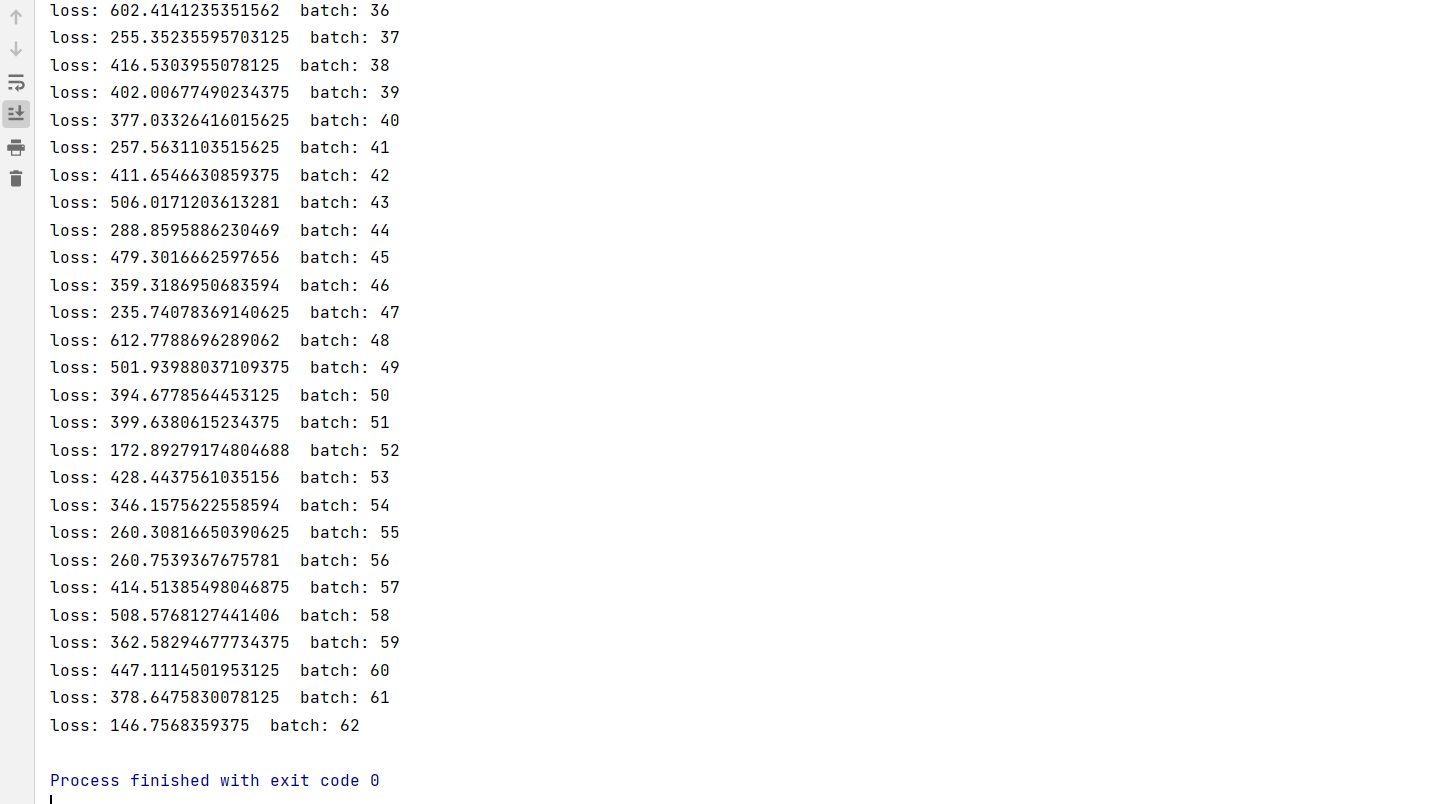
**第三部分：传统机器学习算法设计及应用**（20分）



**代码：**

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
  
*#读取原始数据*X = []  
f = open(**'task3.txt'**,encoding=**'utf-8'**)  
lineIndex = 1  
for v in f:  
 if lineIndex > 1:  
 X.append([float(v.split()[1]), float(v.split()[2])])  
 else:  
 pass  
 lineIndex += 1  
*#转化为numpy array*X= np.array(X)  
  
*#类簇的数量*n\_clusters = 10  
  
*#需要选手补全部分  
#################################################################*keans=KMeans(n\_clusters=10)  
pre=keans.fit(X)  
y=keans.predict(X)  
color=[ **'#F0F8FF'**,**'#FAEBD7'**,**'#FFEBCD'**,**'#DEB887'**,**'#DC143C'**,**'#556B2F'**,**'#8FBC8F'**,**'#00BFFF'**,**'#B22222'**,**'#F0FFF0'**]  
for i in range(len(y)):  
 t=y[i]  
 a=X[i][0]  
 b=X[i][1]  
 plt.scatter(a, b, c=color[t], marker=**'.'**)  
plt.rcParams[**'font.sans-serif'**]=[**'SimHei'**]  
plt.rcParams[**'axes.unicode\_minus'**]=False  
plt.xlabel(**u"经度"**,)  
plt.ylabel(**u"纬度"**,)  
  
  
  
  
*#################################################*plt.title(**'China'**)  
plt.show()

**第四部分：深度学习算法设计及应用**（15分）



*代码：*

*#!/usr/bin/python  
#-\*- coding:cp936 -\*-*

*#epoch=16*import torch  
from torch import nn  
from torch import optim  
import matplotlib.pyplot as plt  
import numpy as np  
import os  
from skimage import io,transform  
import cv2  
import torch.nn.functional as F  
from torchvision import transforms  
from torch.utils.data import DataLoader,Dataset  
from PIL import Image  
device = torch.device(**'cuda:0'**)  
path=**r"D:\label.txt"**class Model(nn.Module):  
 def \_\_init\_\_(self):  
 super(Model, self).\_\_init\_\_()  
 self.conv1 = nn.Conv2d(in\_channels=3, out\_channels=8,kernel\_size=(3,3),stride=1)  
 self.conv2 = nn.Conv2d(in\_channels=8,out\_channels= 16,kernel\_size=(3,3), stride=1)  
 self.conv3 = nn.Conv2d(in\_channels=16,out\_channels=32,kernel\_size=(3,3),stride=1)  
 self.maxpool = nn.MaxPool2d((2, 2), stride=2)  
 self.fc1 = nn.Linear(1152, 128)  
 self.fc2 = nn.Linear(128, 1)  
 def forward(self, x):  
 in\_size = x.size(0)  
  
 out = self.conv1(x)  
 out = F.relu(out)  
 out = self.maxpool(out)  
  
 out = self.conv2(out)  
 out = F.relu(out)  
 out = self.maxpool(out)  
  
 out = self.conv3(out)  
 out = F.relu(out)  
 out = self.maxpool(out)  
  
  
 out = out.view(in\_size, -1)  
  
 out = self.fc1(out)  
 out = F.relu(out)  
  
 out = F.dropout(out, p=0.5)  
 out = self.fc2(out)  
 out = F.relu(out)  
 out = out.squeeze(-1)  
 return out  
class MyDataset(Dataset):  
 def \_\_init\_\_(self, txt\_path):  
 fh = open(txt\_path, **'r'**)  
 imgs = []  
 for line in fh:  
 line = line.rstrip()  
 words = line.split()  
 pic= os.path.join(**r"D:\imgdata"**, words[0])  
 imgs.append((pic, int(words[1])))  
 self.transform = transforms.Compose([  
 transforms.Resize((64, 64)),  
 transforms.ToTensor(),  
 ])  
 self.imgs = imgs  
 def \_\_getitem\_\_(self, index):  
 fn, label = self.imgs[index]  
 img = Image.open(fn).convert(**'RGB'**)  
 img=self.transform(img)  
 return img,label  
 def \_\_len\_\_(self):  
 return len(self.imgs)  
y=[]  
mydata=MyDataset(path)  
dataloader = DataLoader(mydata,batch\_size=16,shuffle=True)  
model=Model().to(device)  
optimizer=optim.Adam(model.parameters(),lr=0.001)  
for batch\_idx, (data, target) in enumerate(dataloader):  
 data, target = data.to(device), target.to(device)  
 optimizer.zero\_grad()  
 output=model(data)  
 loss = F.mse\_loss(output.float(), target.float())  
 loss.backward()  
 optimizer.step()  
 y.append(loss.item())  
 print(**"loss:"**,loss.item(),**" batch:"**,batch\_idx)

**第五部分：人工智能技术综合应用**（20分）

1. 导入相关库

# coding=utf-8

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.neighbors import KNeighborsRegressor

from sklearn.svm import SVR

from sklearn.linear\_model import Lasso

from sklearn.linear\_model import Ridge

from sklearn.neural\_network import MLPRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.tree import ExtraTreeRegressor

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestRegressor

from sklearn.ensemble import AdaBoostRegressor

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.ensemble import BaggingRegressor

1. 数据预处理

scale=StandardScaler()

X\_train=[]

Y\_train=[]

X\_test=[]

Y\_test=[]

df = pd.read\_csv("train.CSV", encoding='GB18030')#读取训练数据

Elevator = np.array(df.iloc[:, 0])#读取第一列

Floor = np.array(df.iloc[:, 1])#读取第列

Layout = np.array(df.iloc[:, 2])#读取第三列

Region = np.array(df.iloc[:, 3])#读取第四列

Renovatior = np.array(df.iloc[:, 4])#读取第五列

data = np.array(df)

Elevator\_set = list(set(Elevator))#把特征转为集合来统计特征矩阵的不同类别

# Floor\_set=set(Floor)

Layout\_set = list(set(Layout))

Region\_set = list(set(Region))

Renovatior\_set = list(set(Renovatior))

for i, dt in enumerate(data):

        #对非数字特征进行编码，由于第2，6，7列时数字，故直接用该值，不需要编码

    x = [Elevator\_set.index(dt[0]), dt[1],

         Layout\_set.index(dt[2]), Region\_set.index(dt[3]),

         Renovatior\_set.index(dt[4]), dt[5], dt[6]]

    y = dt[7]

    X\_train.append(x)#收集16000个训练输入数据

    Y\_train.append(y)#收集16000个训练输出数据

X\_train\_standard=scale.fit\_transform(X\_train)#对训练输入数据进行归一化

1. 训练（选择最优模型）

models=[LinearRegression(),KNeighborsRegressor(),Ridge(),Lasso(),MLPRegressor(alpha=20),DecisionTreeRegressor(),ExtraTreeRegressor(),ExtraTreeRegressor(),RandomForestRegressor(),AdaBoostRegressor(),GradientBoostingRegressor(),BaggingRegressor()]

models\_str=['LinearRegression','KNNRegressor','Ridge','Lasso','MLPRegressor','DecisionTree','ExtraTree','ExtraTreeRegressor()','RandomForest','AdaBoost','GradientBoost','Bagging']

score\_=[]

for name,model in zip(models\_str,models):

    print('开始训练模型：'+name)

    model=model   #建立模型

    model.fit(X\_train,Y\_train)

    y\_pred=model.predict(X\_test)

    score=model.score(X\_test,Y\_test)

    score\_.append(str(score)[:5])

    print(name +' 得分:'+str(score))

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

for name,model in zip(models\_str,models):

    print('开始训练模型：'+name)

    model=model   #建立模型

    model.fit(X\_train\_standard,Y\_train)

    y\_pred=model.predict(X\_test\_standard)

    score=model.score(X\_test\_standard,Y\_test)

    score\_.append(str(score)[:5])

    print(name +' 得分:'+str(score))

运行结果：

开始训练模型：LinearRegression

LinearRegression 得分:0.562813750073091

开始训练模型：KNNRegressor

KNNRegressor 得分:0.7104671149413997

开始训练模型：Ridge

Ridge 得分:0.5628128428599399

开始训练模型：Lasso

Lasso 得分:0.5628382731045105

开始训练模型：MLPRegressor

MLPRegressor 得分:0.6645021274090077

开始训练模型：DecisionTree

DecisionTree 得分:0.6759166399658447

开始训练模型：ExtraTree

ExtraTree 得分:0.6354989943719606

开始训练模型：ExtraTreeRegressor()

ExtraTreeRegressor() 得分:0.5549429772290768

开始训练模型：RandomForest

RandomForest 得分:0.8319795962155914

开始训练模型：AdaBoost

AdaBoost 得分:0.3941078383360318

开始训练模型：GradientBoost

GradientBoost 得分:0.8014437738395972

开始训练模型：Bagging

Bagging 得分:0.8267210825363152

**故选择**RandomForest模型进行预测